ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN TỰ ĐỘNG

----------o0o----------

BÁO CÁO BÀI TẬP LỚN MÔN KỸ THUẬT ROBOT

CHỦ ĐỀ
OPENMANIPULATOR-X ROBOT
USING MATLAB GUIDE

GVHD: NGUYỄN HOÀNG GIÁP

Họ và tên: Nguyễn Quốc Vương

MSSV: 1916000

TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2021

MỤC LỤC

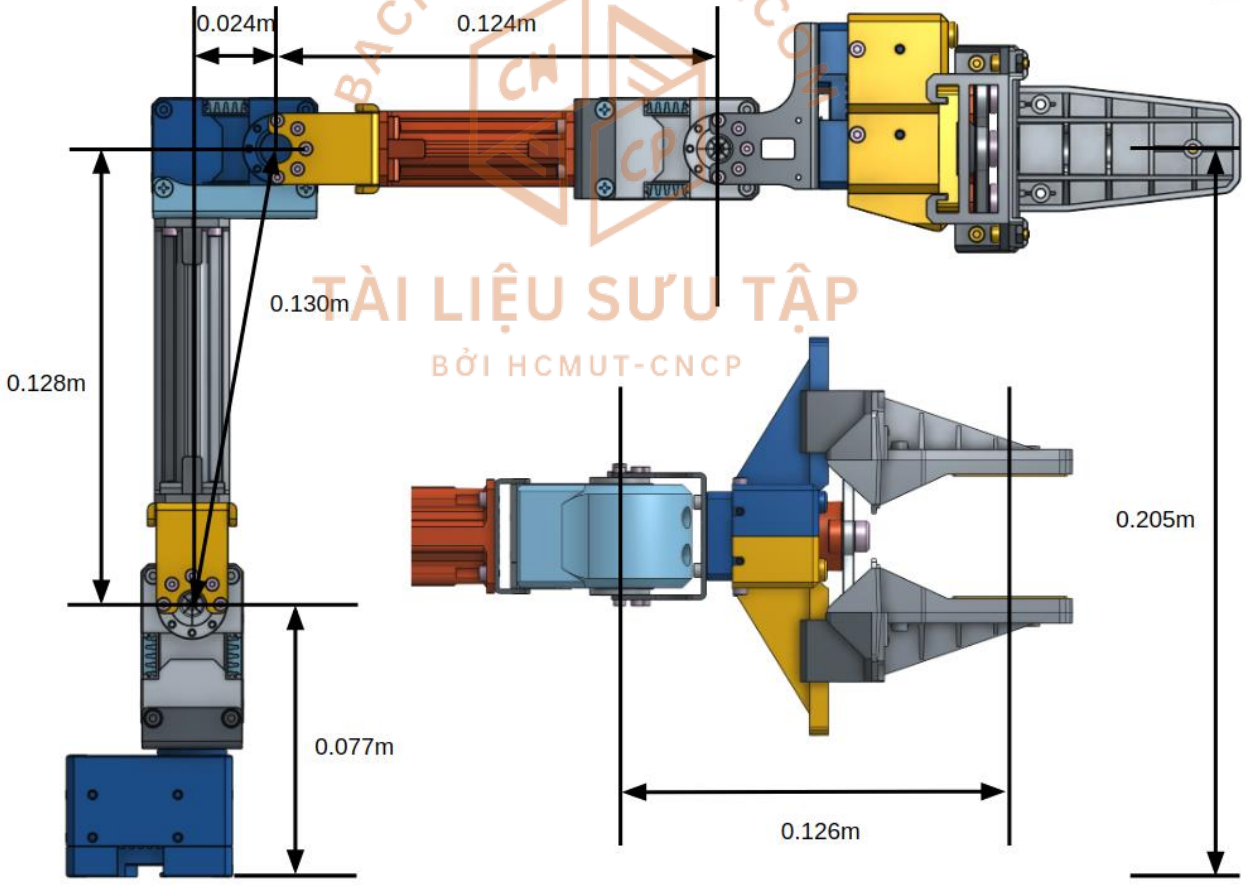
1. Giới thiệu

1.1. Giới thiệu về OPENMANIPULATOR

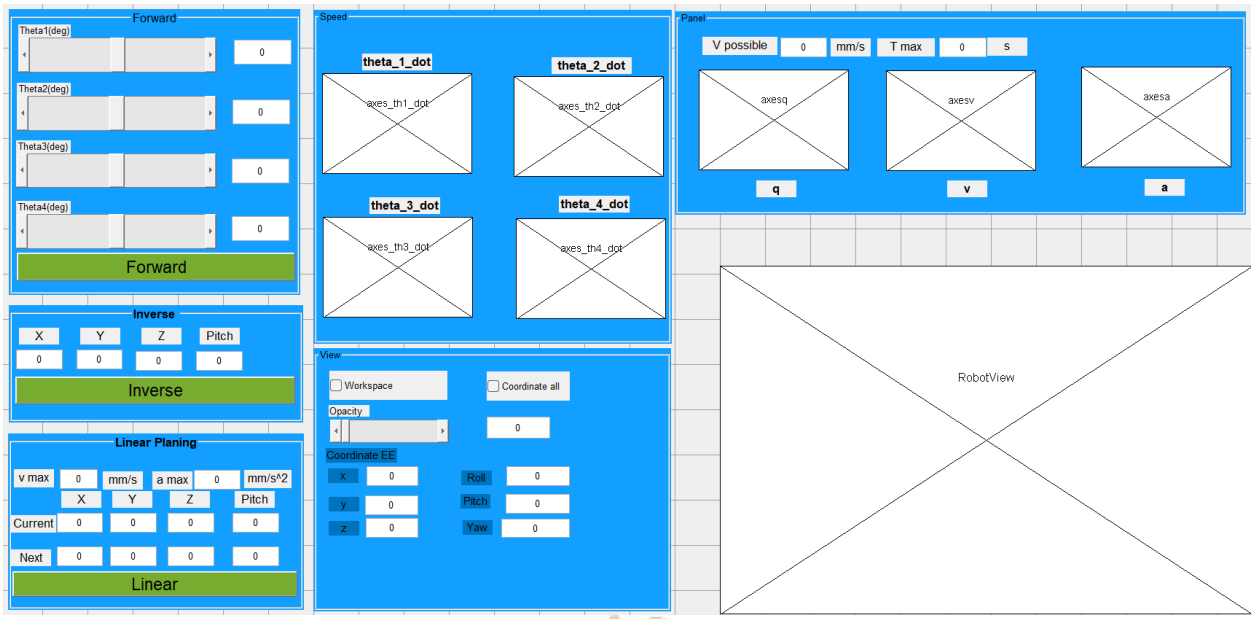
Theo Robotic Institute of America (RIA): Robot là một hệ thống đa tác vụ có thể lập trình được thiết kế để di chuyển vật liệu, bộ phận, dụng cụ hoặc thiết bị chuyên dụng thông qua chuyển động được lập trình theo các biến số để thực hiện các nhiệm vụ cụ thể đó, ngoài ra còn có thể thu thập thông tin từ môi trường và di chuyển thông minh theo đáp ứng.

Nhìn chung, "Robotic" là thuật ngữ khoa học để định nghĩa lĩnh vực nghiên cứu khoa học về sự kết nối thông minh giữa việc ra quyết định và hành động. Do đó, có thể nhận định Robot là một chủ đề liên ngành liên quan đến các lĩnh vực cơ khí, điều khiển, máy tính và điện tử.

Bài tập lớn này sẽ mô phỏng OpenMAINIPULATOR-X ROBOTsử dụng lập trình giao diện MATLAB GUIDE. Các vấn đề chính của bài tập lớn cần được giải quyết bao gồm:



1.2. GUIDE MATLAB



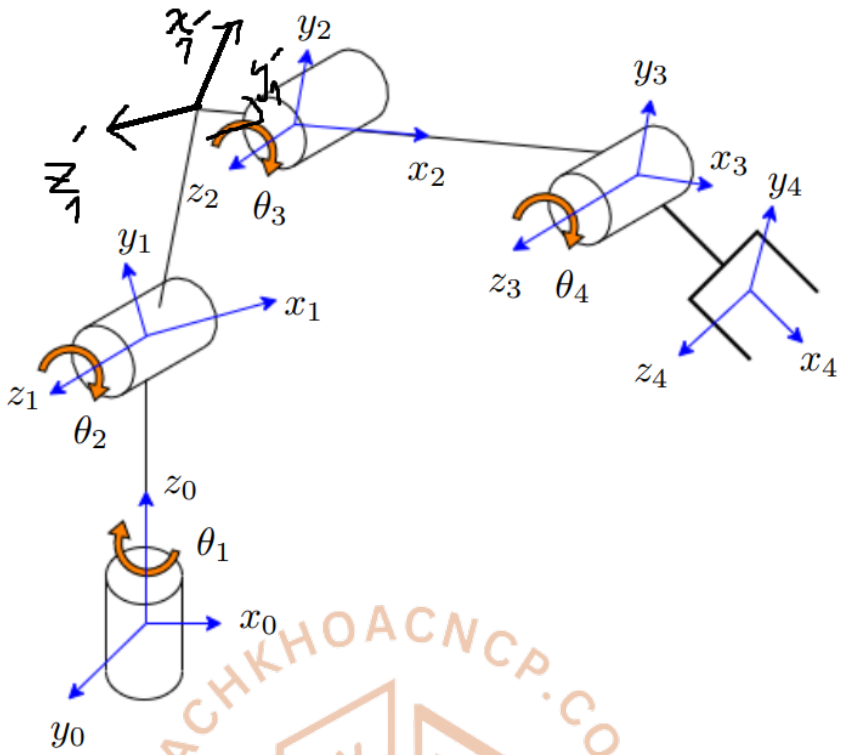
2. Động học thuận

2.1. Lý thuyết và cách thực hiện

2.1.1. Mô phỏng hình dạng robot

Để có thể mô phỏng lại hình dạng robot, ta cần biết được vị trí của các khớp trong hệ toạ độ XYZ từ đó nối các điểm các khớp lại ta sẽ được hình dạng cơ bản của robot OpenManipulator trong không gian.

OpenManipulator là robot 4 bậc tự do, do đó, để có được vị trí của các khớp so với base, ta cần tính các ma trận biến đổi thuận nhất của các khớp so với base, từ đó lấy ra thông số vị trí và hướng của các khớp.



Đầu tiên ta cần thiết lập bảng DH, từ đó có thể thấy, các biến của robot OpenManipulator là θ_1, θ_2, θ_3 và θ_4.

d	θ	a	α
77	theta1	0	90
0	theta2+90	128	0
0	theta3=90		0
0	theta4	126	0

Tiếp theo, ta sẽ tiến hành tính các ma trận biến đổi thuần nhất:

```
A01 =

[ cos(theta_1), 0,  sin(theta_1),  0]
[ sin(theta_1), 0, -cos(theta_1),  0]
[            0, 1,             0, d1]
[            0, 0,             0,  1]


A1s =

[ cos(theta_2 + pi/2), -sin(theta_2 + pi/2), 0, a2*cos(theta_2 + pi/2)]
[ sin(theta_2 + pi/2),  cos(theta_2 + pi/2), 0, a2*sin(theta_2 + pi/2)]
[                   0,                    0, 1,                     0]
[                   0,                    0, 0,                     1]


As2 =

[  0, 1, 0,   0]
[ -1, 0, 0, -a3]
[  0, 0, 1,   0]
[  0, 0, 0,   1]


A23 =

[ cos(theta_3), -sin(theta_3), 0, a4*cos(theta_3)]
[ sin(theta_3),  cos(theta_3), 0, a4*sin(theta_3)]
[            0,             0, 1,               0]
[            0,             0, 0,               1]
```

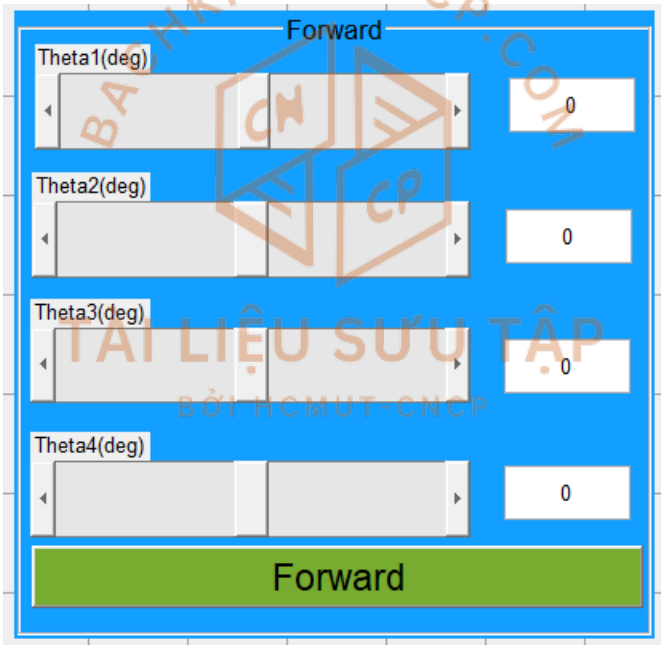

Từ đây ta sẽ tính ra các ma trận ${}^0A_2$, ${}^0A_3$, ${}^0A_4$ để lấy các giá trị vị trí và hướng của các khớp so với base. 10

$$
{}^{B}T_{E} = \begin{bmatrix} c_{124} & s_{124} & 0 & a_1c_1 + a_2c_{12} \\ s_{124} & -c_{124} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & d_1 + d_3 + d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow p = p(\theta_1, \theta_2, d_3)
$$

$$
R(\theta_1, \theta_2, \theta_4) = \begin{bmatrix} n & s & a \end{bmatrix}
$$

Từ các dữ liệu trên ta sẽ tiến hành code matlab

Đầu tiên ta tiến hành thiết kế giao diện GUI:



Xử lý thao tác của nút Forward:

Ta sẽ tiến hành lấy vào các giá trị là θ_1, θ_2, θ_3 và θ_4, sau đó dựa vào các ma trận biến đổi thuần nhất đã tính ➔ Tính toán ra các giá trị vị trí của robot trong không gian

```
global theta1 theta2 theta3 theta4
theta1 = str2double(handles.edit_theta1.String);
theta_1_s = theta1;
```

```
set(handles.slider_theta1,'Value', theta_1_s);
theta2 = str2double(handles.edit_theta2.String);
theta_2_s = theta2;
set(handles.slider_theta2,'Value',theta_2_s);
theta3 = str2double(handles.edit_theta3.String);
theta_3_s = theta3;
set(handles.slider_theta3,'Value',theta_3_s);
theta4 = str2double(handles.edit_theta4.String);
theta_4_s = theta4;
set(handles.slider_theta4,'Value',theta_4_s);
global old_theta_1 old_theta_2 old_theta_3 old_theta_4
oldtheta1 = old_theta_1;
oldtheta2 = old_theta_2;
oldtheta3 = old_theta_3;
oldtheta4 = old_theta_4;
for k = 1:30
    theta1_temp = old_theta_1 + (theta1 - oldtheta1)*k/30;
    theta2_temp = old_theta_2 + (theta2 - oldtheta2)*k/30;
    theta3_temp = old_theta_3 + (theta3 - oldtheta3)*k/30;
    theta4_temp = old_theta_4 + (theta4 - oldtheta4)*k/30;
    DrawRobot(theta1_temp, theta2_temp, theta3_temp, theta4_temp, handles);


   set(handles.btn_foward,'Enable','off','String', 'Not Push');
   pause(0.2);
end
set(handles.btn_foward,'Enable','on','String', 'Foward');
```

Sau đó ta sẽ tiến hành vẽ cánh tay robot bằng lệnh Plot3 (nối các điểm vị trí giữa các khớp lại với nhau và tạo ra một điểm trong không gian để xác định các khớp). Tiếp theo, dựa vào ma trận biến đổi thuần nhất, sử dụng lệnh quiver3 để vẽ hướng của các khớp so với base.

```
[X,Y,Z] = cylinder(r);
h = 35;
Z = Z*h;
surf(X,Y,Z)
%     x0 = [23 -46 -46 23 23 -46 -46 23];
```

```
%     y0 = [23 23 -23 -23 23 23 -23 -23];
% %     x0 = [23 0 -10 -46 -46 -10 0 23 23 0 -10 -46 -46 -10 0 23];
% %     y0 = [23 23 23 23 -23 -23 -23 -23 23 23 23 23 -23 -23 -23 -23];
%     z0 = [0 0 0 0 40 40 40 40 ];
%     boxplot(x0,y0,z0,col, opacity);
 %link 1
x_link1=[0 A01(1,4)];
y_link1=[0 A01(2,4)];
z_link1=[0 A01(3,4)];
plot3(x_link1,y_link1,z_link1,'Linewidth',5,'color',[0.8 0.8 1 opacity ]);hold
on;grid on;
scatter3( A01(1,4),A01(2,4),A01(3,4), 'SizeData',80,
'MarkerFaceColor','m','MarkerEdgeColor','m','MarkerFaceAlpha',opacity,'Marker
EdgeAlpha',opacity);

%link 2
x_link2=[A01(1,4) A02(1,4)];
y_link2=[A01(2,4) A02(2,4)];
z_link2=[A01(3,4) A02(3,4)];
scatter3( A02(1,4),A02(2,4),A02(3,4), 'SizeData',80,
'MarkerFaceColor','m','MarkerEdgeColor','m','MarkerFaceAlpha',opacity,'Marker
EdgeAlpha',opacity);
plot3(x_link2,y_link2,z_link2,'Linewidth',5,'color',[0.8 0.8 1 opacity ]);hold
on;grid on;

%link 2p
x_link2p=[A02(1,4) A03(1,4)];
y_link2p=[A02(2,4) A03(2,4)];
z_link2p=[A02(3,4) A03(3,4)];
scatter3( A03(1,4),A03(2,4),A03(3,4), 'SizeData',80,
'MarkerFaceColor','m','MarkerEdgeColor','m','MarkerFaceAlpha',opacity,'Marker
EdgeAlpha',opacity);
plot3(x_link2p,y_link2p,z_link2p,'Linewidth',5,'color',[0.8 0.8 1 opacity ]);hold
on;grid on;

%link 3
x_link3=[A03(1,4) A04(1,4)];
y_link3=[A03(2,4) A04(2,4)];
z_link3=[A03(3,4) A04(3,4)];
```

```
scatter3( A04(1,4),A04(2,4),A04(3,4), 'SizeData',80,
'MarkerFaceColor','m','MarkerEdgeColor','m','MarkerFaceAlpha',opacity,'Marker
EdgeAlpha',opacity);
plot3(x_link3,y_link3,z_link3,'Linewidth',5,'color',[0.8 0.8 1 opacity ]);hold
on;grid on;

%link 4
x_link4=[A04(1,4) A05(1,4)];
y_link4=[A04(2,4) A05(2,4)];
z_link4=[A04(3,4) A05(3,4)];
scatter3( A05(1,4),A05(2,4),A05(3,4), 'SizeData',80,
'MarkerFaceColor','m','MarkerEdgeColor','m','MarkerFaceAlpha',opacity,'Marker
EdgeAlpha',opacity);
plot3(x_link4,y_link4,z_link4,'Linewidth',5,'color',[0.8 0.8 1 opacity ]);hold
on;grid on;
```

Cuối cùng, ta đưa tất cả code trên vào hàm draw để dễ sử dụng và quản lý.

- function drawRobot(theta1, theta2, theta3, theta4, handles)

2.1.2. Các thông số vị trí và RPY của EF vector

Ta sẽ tiến hành lấy các thông số vị trí và hướng của EF vector dựa vào ma trận biến đổi thuần nhất ${}^0A_4$

$$ {}^BT_E = {}^0A_1(q_1)\,{}^1A_2(q_2)\,{}^2A_3(q_3)\,{}^3A_4(q_4) = {}^0A_4(q_1,q_2,q_3,q_4) $$

$$ {}^BT_E = \begin{bmatrix} c_{124} & s_{124} & 0 & a_1c_1 + a_2c_{12} \\ s_{124} & -c_{124} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & d_1 + d_3 + d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow p = p(\theta_1,\theta_2,d_3) $$

$$ \Downarrow $$

$$ R(\theta_1,\theta_2,\theta_4) = \begin{bmatrix} n & s & a \end{bmatrix} $$

Với cách tính RPY như sau:

```
Ox5 = A05(1,4); Oy5 = A05(2,4);Oz5 = A05(3,4);
tmp = A05(1:3,1:3);
calR_P_Y = CaculateR_P_Y(tmp);
set(handles.edit_x,'String', num2str(round(Ox5,2)));
set(handles.edit_y,'String', num2str(round(Oy5,2)));
set(handles.edit_z,'String', num2str(round(Oz5,2)));
set(handles.edit_roll,'String',num2str(round(calR_P_Y(1),2)));
set(handles.edit_pitch,'String',num2str(round(calR_P_Y(2),2)));
set(handles.edit_yaw,'String',num2str(round(calR_P_Y(3),2)));

function [res] = CaculateR_P_Y(T)
   cal = sqrt(T(3,2)^2 + T(3,3)^2);
   pitch = atan2(-T(3,1),cal);
   if pitch == pi/2
      yaw = 0;
      roll = atan2(T(1,2),T(2,2));
   elseif pitch == -pi/2
      yaw = 0;
      roll = -atan2(T(1,2),T(2,2));
   else
      yaw = atan2(T(2,1)/cos(pitch), T(1,1)/cos(pitch));
      roll = atan2(T(3,2)/cos(pitch), T(3,3)/cos(pitch));
   end
   res = [roll pitch yaw]*(180/pi);
end
```

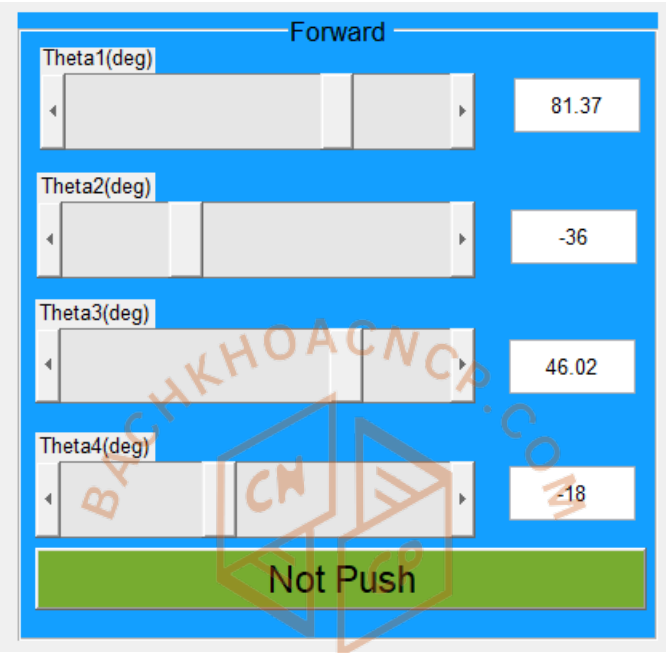
Để có thể xoay được đồ thị ta dùng lệnh.

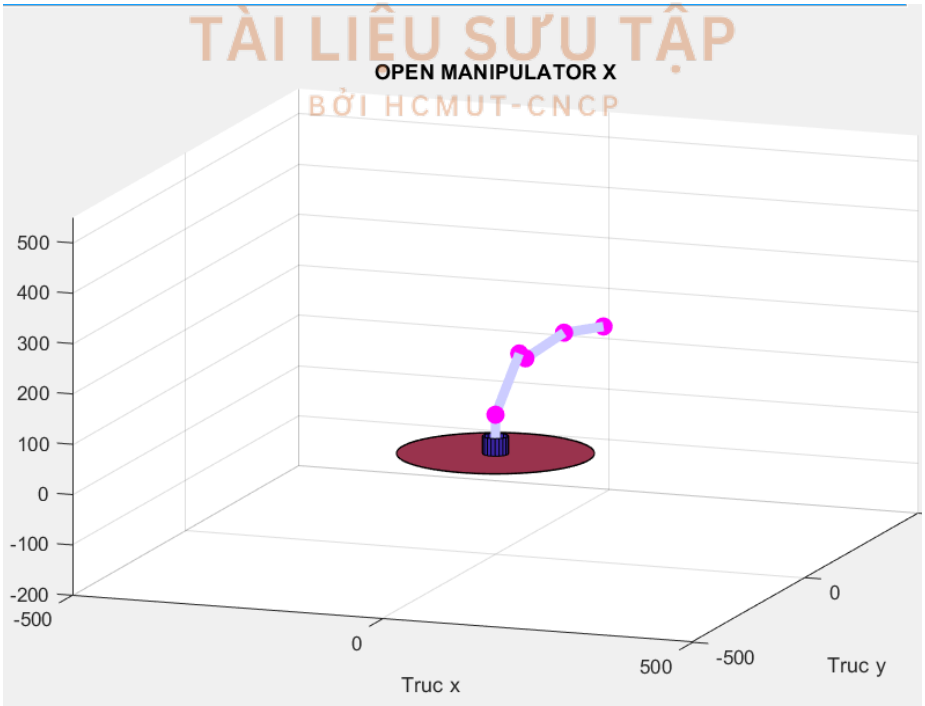
```
%xoay
h = rotate3d;
h.Enable = 'on';
```

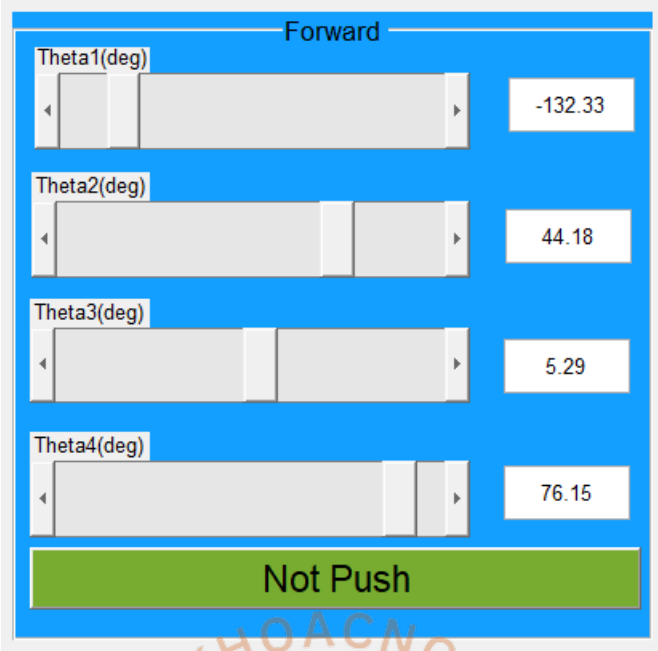
Tương tự ở trên, ta tạo hàm coordinate() để dễ sử dụng và quản lý

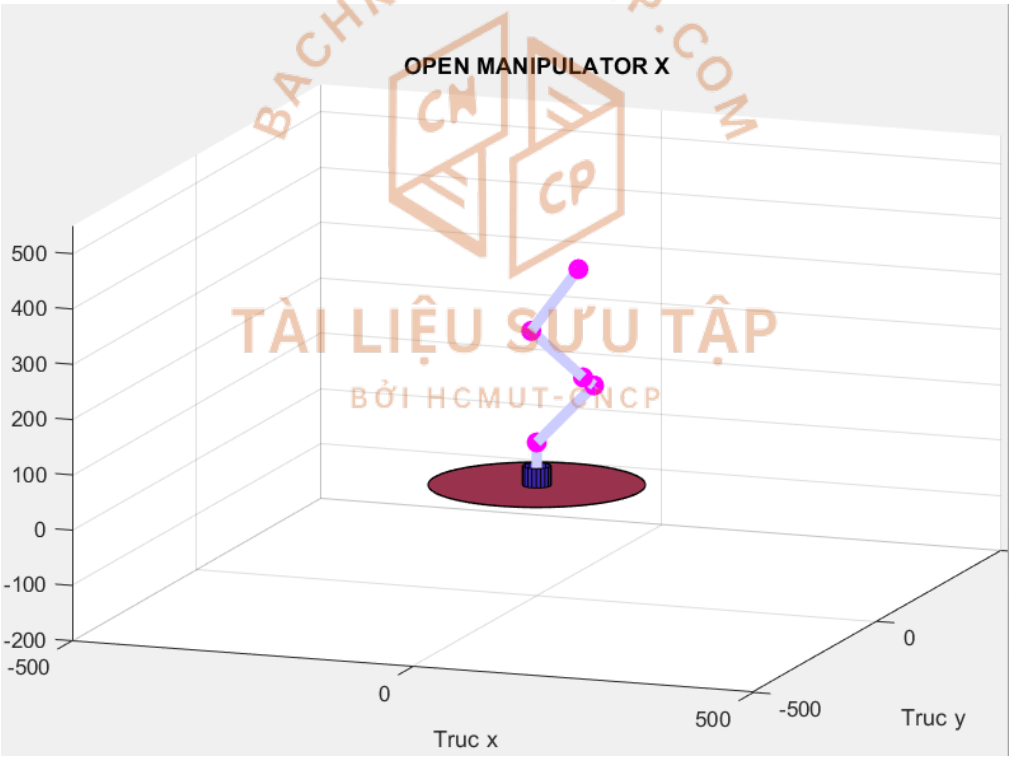
- function coordinate(theta1, theta2, theta3, theta4,handles)

2.2. Kết quả thực hiện









3. Động học ngược

3.1. Lý thuyết và cách thực hiện

Ta sẽ tính toán các công thức động học ngược robot Openmanipulator, theo các bước sau

1) Xác định bảng DH

Joint	θ_i (°)	α_i (°)	a_i (m)	d_i (m)
1	θ_1	90	0	0.077
2	$\theta_1 - \theta_0$	0	0.130	0
3	$\theta_3 + \theta_0$	0	0.135	0
4	θ_4	0	0.126	0

2) Áp dụng công thức động học thuận. Xây dựng ma trận biến đổi thuận nhất ${}^0A_4$

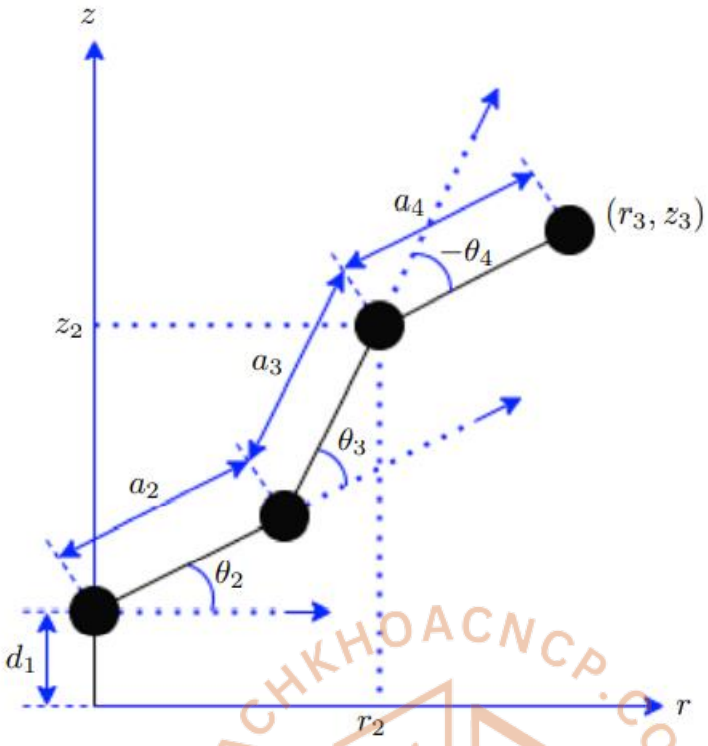
$$ {}^BT_E = {}^0A_1(q_1)\,{}^1A_2(q_2)\,{}^2A_3(q_3)\,{}^3A_4(q_4) = {}^0A_4(q_1,q_2,q_3,q_4) $$

$$ {}^BT_E = \begin{bmatrix} c_{124} & s_{124} & 0 & a_1c_1 + a_2c_{12} \\ s_{124} & -c_{124} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & d_1 + d_3 + d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow p = p(\theta_1,\theta_2,d_3) $$

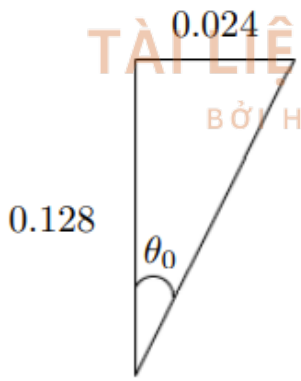
$$ R(\theta_1,\theta_2,\theta_4) = [n \quad s \quad a] $$

3) Xác định Joint space/ Tool Space

Trong bài robot OPENMANIPULATOR, ta chọn Tool Space là các giá trị vị trí X,Y,Z và góc Picth để tìm các biến là θ_1, θ_2, θ_3 và θ_4.



4) Dựa vào các công thức lượng giác. Xây dựng công thức Inverse



$$\theta_0 = tan^{-1}\left(\frac{0.024}{0.128}\right) \approx 11°$$

$$\phi = \theta_2 + \theta_3 + \theta_4$$

$$r_2 = r_3 - a_4 \ cos \ \phi$$

$$z_2 = z_3 - a_4 \ sin \ \phi$$

$$cos \ \theta_3 = \frac{{r_2}^2 + {z_2}^2 - \left({a_2}^2 + {a_3}^2\right)}{2 \ a_2 \ a_3}$$

$$\theta_3 = \pm \ cos^{-1}\left(\frac{{r_2}^2 + {z_2}^2 - \left({a_2}^2 + {a_3}^2\right)}{2 \ a_2 \ a_3}\right)$$

$$\theta_3 = \pm\ cos^{-1}\left(\frac{{r_2}^2 + {z_2}^2 - \left({a_2}^2 + {a_3}^2\right)}{2\ a_2\ a_3}\right)$$

$$r_2 = a_2\ cos\ \theta_2 + a_3\ cos\left(\theta_2 + \theta_3\right)$$

$$z_2 = a_2\ sin\ \theta_2 + a_3\ sin\left(\theta_2 + \theta_3\right)$$

$$r_2 = cos\ \theta_2\left(a_2 + a_3\ cos\ \theta_3\right) - sin\ \theta_2\left(a_3\ sin\ \theta_3\right)$$

$$z_2 = cos\ \theta_2\left(a_3\ sin\ \theta_3\right) + sin\ \theta_2\left(a_2 + a_3\ cos\ \theta_3\right)$$

$$cos\ \theta_2 = \frac{\left(a_2 + a_3\ cos\ \theta_3\right) r_2 + \left(a_3\ sin\ \theta_3\right) z_2}{{r_2}^2 + {z_2}^2}$$

$$sin\ \theta_2 = \frac{\left(a_2 + a_3\ cos\ \theta_3\right) z_2 + \left(a_3\ sin\ \theta_3\right) r_2}{{r_2}^2 + {z_2}^2}$$

$$\theta_2 = tan^{-1}\left(\frac{sin\ \theta_2}{cos\ \theta_2}\right)$$

$$\theta_4 = \phi - \left(\theta_2 + \theta_3\right)$$

Tiến hành code cho phần inverse

Ta tạo hàm Inverse để tính toán động học ngược theo các thông số X, Y, Z, Yall được nhập vào

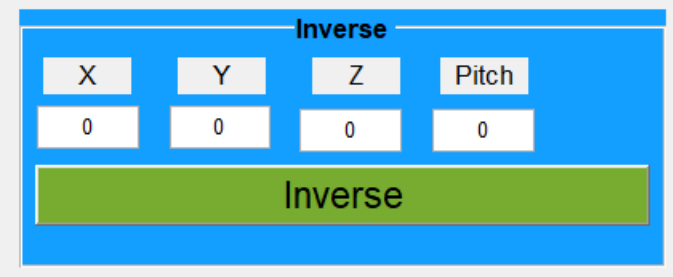
```
function [res, theta_inv] = Check_Inverse(handles, oldtheta2, oldtheta3)
    x = str2double(handles.Pos_X.String);
    y = str2double(handles.Pos_Y.String);
    z = str2double(handles.Pos_Z.String);
    phi= -str2double(handles.Pitch.String)*pi/180;
```

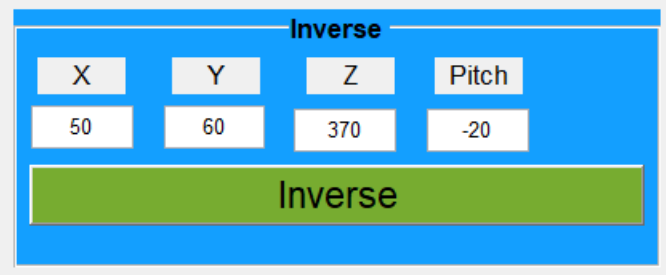
```
    theta_1 = round(atan2(y,x),4);

    pr = round(sqrt(x^2+y^2),4);
    r3 = pr;
    z3 = z - 77;
    a4 = 126;
    a2 = 130;
    a3 = 124;
    r2 = round(r3 - a4*cos(phi),4);
    z2 = round(z3 - a4*sin(phi),4);
%     delta_theta_1 = atan(24/128)
    delta_theta = atan(24/128);
%     (r2^2 +z2^2-(a2^2 +a3^2))/(2*a2*a3)

    if (round((r2^2 +z2^2-(a2^2 +a3^2))/(2*a2*a3),2) >1)
        res = 1;
        theta_inv = zeros(4,1);
        return
    else
       c_theta_3 = round(round((r2^2 +z2^2-(a2^2 +a3^2))/(2*a2*a3),2),4);
       s_theta_3 = [round(-sqrt(1-c_theta_3^2),4); round(sqrt(1-c_theta_3^2),4)];
       theta_3_cal_matrix = round(atan2(s_theta_3, c_theta_3),4);
    end

    theta_2_cal_matrix = zeros(2,1);
    for n = 1:2
%         ctheta_2 = ( ( a2 + a3*cos(theta_3_matrix(n,:) )*r2 + (
a3*sin(theta_3_matrix(n,:) )*z2 )))/(r2^2 +z2^2);
%         stheta_2 = ((a2 + a3*cos(theta_3_matrix(n,:))*z2 +
(a3*sin(theta_3_matrix(n,:)))*r2))/(r2^2 +z2^2);
        ctheta_2 = round(((a2 + a3*cos(theta_3_cal_matrix(n,:)))*r2 +
a3*sin(theta_3_cal_matrix(n,:))*z2)/(r2^2+z2^2),4);
        stheta_2 = round(((a2 + a3*cos(theta_3_cal_matrix(n,:)))*z2 -
a3*sin(theta_3_cal_matrix(n,:))*r2)/(r2^2+z2^2),4);
        theta_2_cal_matrix(n,:) = round(atan2(stheta_2,ctheta_2),4);
    end
    theta_2_matrix = theta_2_cal_matrix + delta_theta - pi/2;
    theta_3_matrix = theta_3_cal_matrix  - delta_theta + pi/2;
```

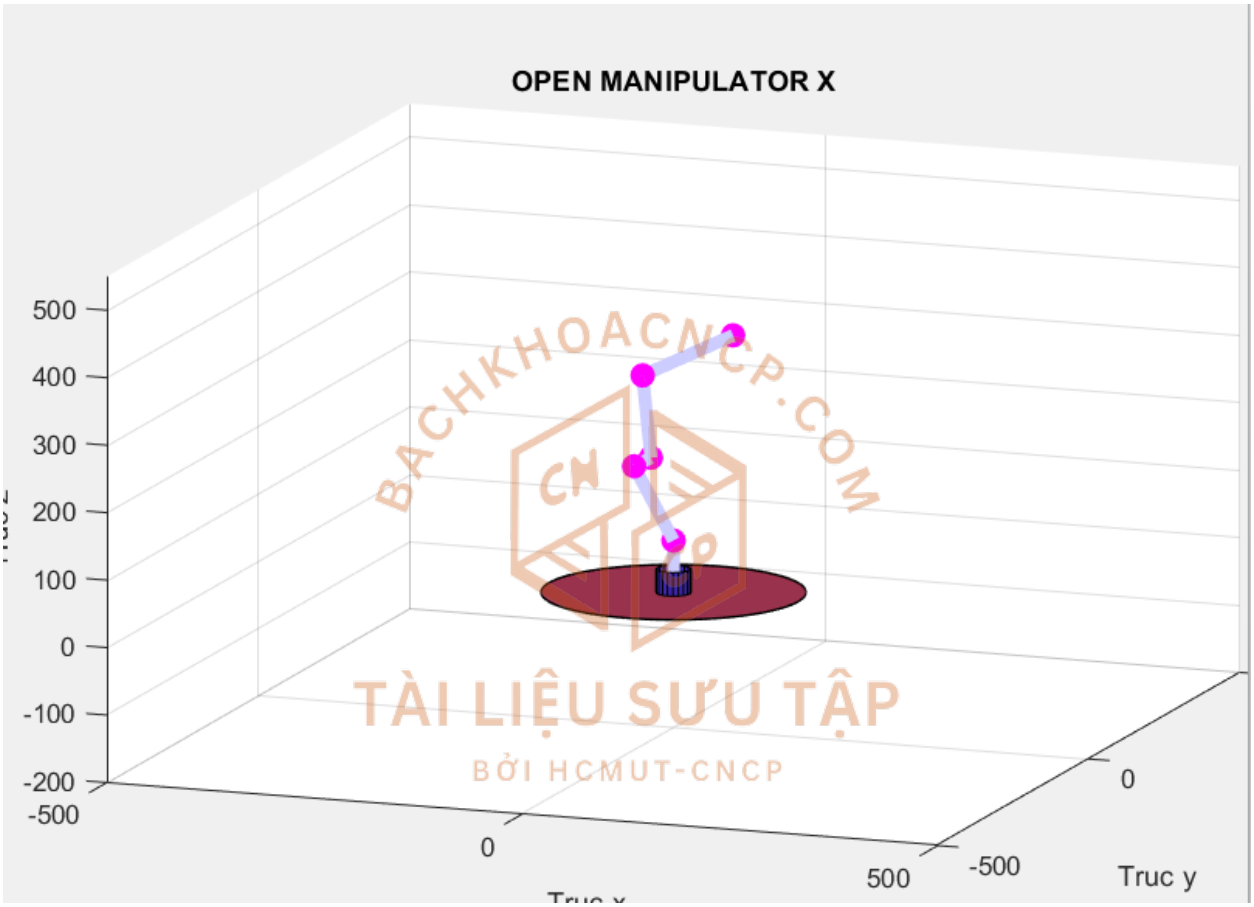
```
%       for n = 1:2
%         if(theta_2_matrix(n,:)<-pi/2 ||(theta_2_matrix(n,:)>
pi/2)||(theta_3_matrix(n,:))<-pi/2||(theta_3_matrix(n,:))> pi/2)
%             theta_2_matrix(n,:) = 0;
%             theta_3_matrix(n,:) = 0;
%         end
%     end
    flag = zeros(2,1);
    if(theta_2_matrix(1,1) <-pi/2 || theta_2_matrix(1,1) > pi/2 ||
theta_3_matrix(1,1) <-pi/2 || theta_3_matrix(1,1) > pi/2)
        flag(1,1) = 1;
    end
    if(theta_2_matrix(2,1) <-pi/2 || theta_2_matrix(2,1) > pi/2 ||
theta_3_matrix(2,1) <-pi/2 || theta_3_matrix(2,1) > pi/2)
        flag(2,1) = 1;
    end
%     flag
    if (flag(1,1) == 1 && flag(2,1) ==1)
       res = 1;
        theta_inv = zeros(4,1);
        return
    elseif (flag(1,1) ==1 && flag(2,1) == 0)
        theta_2 = theta_2_matrix(2,1);
        theta_3 = theta_3_matrix(2,1);
        note = 0;
    elseif(flag(2,1) ==1 && flag(1,1) == 0)
        theta_2 = theta_2_matrix(1,1);
        theta_3 = theta_3_matrix(1,1);
        note = 0;
    else
        note = 1;
    end
    if(note)
        delta1 =  abs(oldtheta2 - theta_2_matrix(1,1))+ abs(oldtheta3 -
theta_3_matrix(1,1));
        delta2 =  abs(oldtheta2 - theta_2_matrix(2,1))+ abs(oldtheta3 -
theta_3_matrix(2,1));
        if(delta1 < delta2)
            theta_2 = theta_2_matrix(1,1);
            theta_3 = theta_3_matrix(1,1);
```

```
        else
            theta_2 = theta_2_matrix(2,1);
            theta_3 = theta_3_matrix(2,1);
        end
    end
    theta_4 = phi - theta_2 - theta_3;


    if (round(theta_1,4) < round(-pi,4) || round(theta_1,4) > round(pi,4))
       res = 1;
        theta_inv = zeros(4,1);
        return
    end

    if (theta_4 < -pi/2 || theta_4 > pi/2)
       res = 1;
        theta_inv = zeros(4,1);
        return
    end
    res= 0;

    theta_inv = [theta_1; theta_2; theta_3; theta_4];
end
```


3.2. Kết quả thực hiện







4. Tạo Animation cho robot và vẽ WorkSpace

4.1. Animation

4.1.1. Lý thuyết và cách thực hiện

Vòng lặp **for** là một cấu trúc điều khiển lặp đi lặp lại một vòng lặp được lập trình để thực hiện trong một khoảng thời gian cụ thể nào đó.

Do vậy, để thực hiện tạo chuyển động giả lập cho robot, ta sử dụng vòng lặp **for**, với số lần lặp phụ thuộc vào độ mượt của chuyển động.

Độ mượt của chuyển động ở đây lại phụ thuộc vào số khoảng chia vị trí của robot khi di chuyển từ vị trí A sang vị trí B.

Tại vị trí A, ta sẽ có các thông số DH-variable (góc khớp, d3), tạm gọi là thông số DH_old. Tương tự, tại vị trí B, ta sẽ có các thống số DH_new. Bằng cách sử dụng các biến global ta sẽ lấy được các giá trị DH-variable.

```
% ==== lay du lieu tu GUI ===
global theta1 theta2 d3 theta4 t1_old t2_old d3_old t4_old;

t1_old = theta1;
t2_old = theta2;
d3_old = d3;
t4_old = theta4;
[theta1, theta2, d3, theta4] = getdata(handles);
```

Tiếp theo, ta sẽ tiến hành chia các khoảng giá trị của các thông số DH để tạo thành các vị trí của robot giữa 2 điểm A và B. Ở đây, em chia thành 10 khoảng

```
%% chia vi tri
dtheta1 = (theta1 - t1_old)/20;
dtheta2 = (theta2 - t2_old)/20;
dd3 = (d3 - d3_old)/20;
dtheta4 = (theta4 - t4_old)/20;
```

Cuối cùng, tạo vòng **for** để vẽ robot ở 20 vị trí trong không gian. Ở đây, ta cần thêm trễ 0.001s vào trong vòng **for** để có thể nhìn thấy chuyển động của robot

```
for i = 1:20
    theta1 = t1_old + i*dtheta1;
    theta2 = t2_old + i*dtheta2;
    d3 = d3_old + i*dd3;
    theta4 = t4_old + i*dtheta4;

d3 = -d3;
draw(theta1, theta2, d3, theta4, handles);
coordinate(theta1, theta2, d3, theta4, handles);
pause(0.001);
```

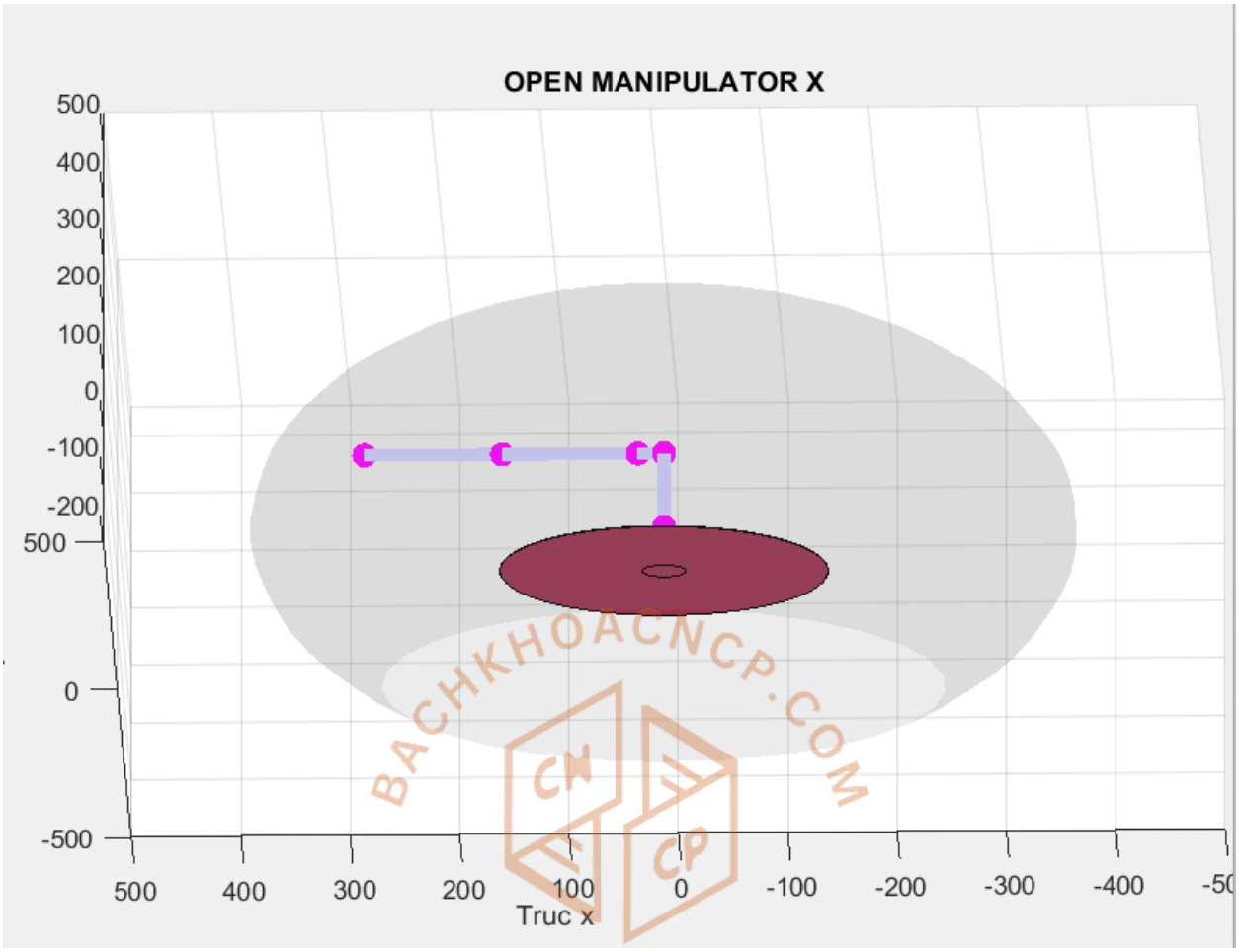
```
end
```

4.2. WorkSpace

4.2.1. Lý thuyết và cách thực hiện

```
function Draw_Workspace()
hold on
r = sqrt(378^2+24^2);
[u,v] = meshgrid(linspace(0,2*pi,30),linspace(0,pi,30));
x = r.*cos(u).*sin(v);
y = r.*sin(u).*sin(v);
z = r.*cos(v)+77;
color = [0.5 0.5 0.5];
surf(x,y,z,'EdgeColor','none','Facecolor',color,'Facealpha',0.15);

end
```


4.2.2. Kết quả thực hiện

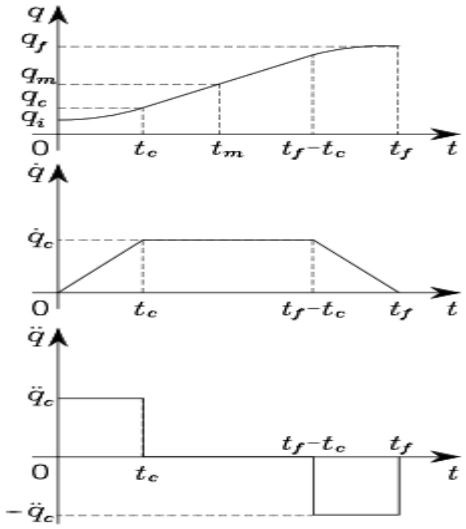


5. Quy hoạch quỹ đạo Trajectory Planning

5.1. SSLý thuyết và cách thực hiện

Hoạch định quỹ đạo, quy hoạch quỹ đạo nghĩa là xác định quy luật chuyển động các biến khớp để điều khiển chuyển động từng khớp và tổng hợp chúng thành chuyển động của cả robot theo một quỹ đạo xác định.

Ta thường sử dụng phương pháp xấp xỉ các đa thức bậc n, trong bài này ta thực hiện tương đa thức bậc 3. Từ ví dụ như hình sau, tiến hành tính toán và lập trình cho robot.



Trước nhất, ta chia các đoạn từ 0 – t1, t1-t2, t3-t4, t4-t5 để dễ dàng cho việc tìm đa thức bậc 3 theo từng đoạn.

Dễ dàng, tính được t1 = vmax/ amax và t2 = 2t1. Tương tự tìm được, t3 = t2 + (pmax-2vmax*t1)/vmax. Cuối cùng, t4 = t3 + t1 theo tính đối xứng và t5 = t4+ t1.

Sau đó, ta thực hiện tìm các thông số còn lại, và hoàn thiện các đa thức bậc 3.

Tìm điều kiện ràng buộc cho vmax, cuối cùng tiến hành code và chạy debug riêng sau đó đưa vào guide để tiến hành demo

```
for k = 1:K
    if t(k) <t1
        q(k) = amax*(t(k))^2/2;
        v(k)=amax*t(k);
        a(k)=amax;
    elseif t(k) <t2
        q(k)=vmax*t(k)+ amax*t1^2/2 - vmax*t1;
        v(k)=vmax;
        a(k)=0;
    elseif t(k)<=tmax
        q(k) = -1/2*amax*(t(k))^2+amax*t(k)*tmax+qmax-1/2*amax*tmax^2;
        v(k) = amax*(tmax-t(k));
       % v(k) = amax*t1-amax*(t(k)-t2);
        a(k) = -amax;
    end
    x = x_0+(q(k)/qmax)*(x_1-x_0);
    y = y_0+(q(k)/qmax)*(y_1-y_0);
    z = z_0+(q(k)/qmax)*(z_1-z_0);
    pitch = pitch_0 + (q(k)/qmax)*(pitch_1-pitch_0);
    X=[X, x];
    Y=[Y, y];
```

```
        Z=[Z, z];
        Pitch = [Pitch, pitch];
%           x_dot = (v(k)/qmax)*(x_1 - x_0);
%           y_dot =  (v(k)/qmax)*(y_1 - y_0);
%           z_dot = (v(k)/qmax)*(z_1 - z_0);
%           pitch_dot =  (v(k)/qmax)*(pitch_1-pitch_0);
        if k ==1
            x_dot = (x-x_pre)/t(1);
            y_dot = (y-y_pre)/t(1);
            z_dot = (z-z_pre)/t(1);
            pitch_dot = (pitch-pitch_pre)/t(1);
        else
            x_dot = (x - x_pre)/(t(k)-t(k-1));
            y_dot = (y - y_pre)/(t(k)-t(k-1));
            z_dot = (z - z_pre)/(t(k)-t(k-1));
            pitch_dot = (pitch - pitch_pre)/(t(k)-t(k-1));
        end
        X_dot = [X_dot, x_dot];
        Y_dot = [Y_dot,y_dot];
        Z_dot = [Z_dot,z_dot];
        Pitch_dot = [Pitch_dot,pitch_dot];
        x_pre = x;
        y_pre = y;
        z_pre = z;
        pitch_pre = pitch;
    end

    for k = 1:K
         old_theta_1 = theta1;
         old_theta_2 = theta2;
         old_theta_3 = theta3;
         old_theta_4 = theta4;

         v_end = [X_dot(1,k); Y_dot(1,k);Z_dot(1,k);(Pitch_dot(1,k)*pi/180)];
        [flag3 , theta_inv] =
Check_Inverse_Jaco_new(X(1,k),Y(1,k),Z(1,k),Pitch(1,k),old_theta_2,
old_theta_3);
        if flag3 == 0
            theta1 = theta_inv(1,1)*180/pi;
            th_1(1,k) = theta1;
```

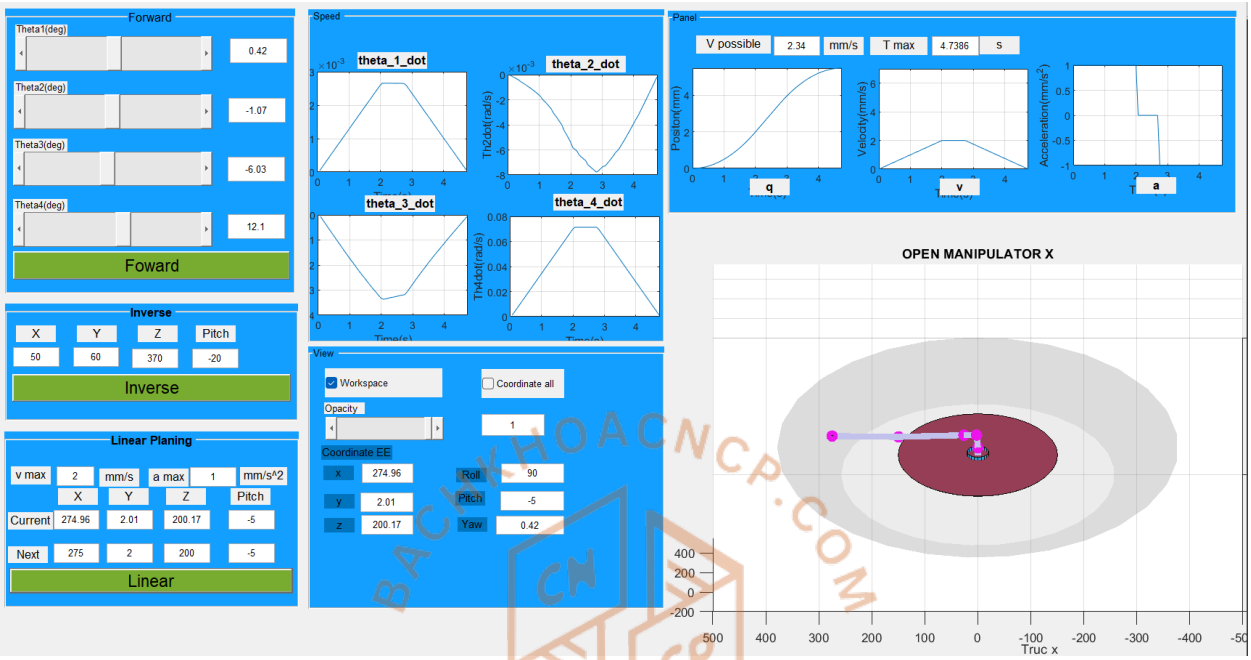
```
            old_theta_1 = theta1;
            theta2 = theta_inv(2,1)*180/pi;
            th_2(1,k) = theta2;
            old_theta_2 = theta2;
            theta3 = theta_inv(3,1)*180/pi;
            th_3(1,k) = theta3;
            old_theta_3 = theta3;
            theta4 = theta_inv(4,1)*180/pi;
            th_4(1,k) = theta4;
            old_theta_4 = theta4;
%               J_Matrix = Jacobian_Matrix(theta_inv(1,1), theta_inv(2,1),
theta_inv(3,1), theta_inv(4,1));
            yaw = Foward_Kinematics(deg2rad(theta1), deg2rad(theta2),
deg2rad(theta3), deg2rad(theta4));
            v_end(4,1) = v_end(4,1)*cos(yaw);
            J_Matrix = Jacobian_Matrix(deg2rad(theta1), deg2rad(theta2),
deg2rad(theta3), deg2rad(theta4));
            v_joint = J_Matrix\v_end;
            v_th1(1,k) = v_joint(1,1);
            v_th2(1,k) = v_joint(2,1);
            v_th3(1,k) = v_joint(3,1);
            v_th4(1,k) = v_joint(4,1);

        else
            msgbox('Out of Workspace','info','error');
            break
        end
    end
    if flag3 == 0
    for k = 1:K

        DrawRobot(th_1(1,k), th_2(1,k),th_3(1,k),th_4(1,k),handles);
%         set(handles.edit_test,'string',num2str(round(-th_2(1,k)-th_3(1,k)-
th_4(1,k),2)));
        hold on;
        plot3(X(1,1:k),Y(1,1:k),Z(1,1:k),'--','linewidth',2,'color','black');
        hold off;
        set(handles.slider_theta1,'Value', th_1(1,k));
        set(handles.edit_theta1,'String', num2str(round( th_1(1,k),2)));
        set(handles.slider_theta2,'Value', th_2(1,k));
```

```
        set(handles.edit_theta2,'String', num2str(round(th_2(1,k),2)));
        set(handles.slider_theta3,'Value', th_3(1,k));
        set(handles.edit_theta3,'String', num2str(round(th_3(1,k),2)));
        set(handles.slider_theta4,'Value', th_4(1,k));
        set(handles.edit_theta4,'String', num2str(round(th_4(1,k),2)));

        plot(handles.axesq,t(1:k),q(1:k));
%         axis(handles.axesq,[0 tmax, 0 qmax])
        xlabel(handles.axesq,'Time(s)')
        ylabel(handles.axesq,'Positon(mm)')
        grid (handles.axesq, 'on')

        plot(handles.axesv,t(1:k), v(1:k));
%         axis(handles.axesv,[0 tmax, 0 vmax])
         xlabel(handles.axesv,'Time(s)')
        ylabel(handles.axesv,'Velocity(mm/s)')
        grid (handles.axesv, 'on')

        plot(handles.axesa,t(1:k), a(1:k));
%         axis(handles.axesa,[0 tmax, -amax amax])
         xlabel(handles.axesa,'Time(s)')
        ylabel(handles.axesa,'Acceleration(mm/s^2)')
        grid (handles.axesa, 'on')

%         plot(handles.axes_th1,t(1:k), th_1(1,1:k));
% %         axis(handles.axes_th1,[0 tmax, -180 180]);
%         xlabel(handles.axes_th1,'Time(s)')
%         ylabel(handles.axes_th1,'Theta1(deg)')
%         grid (handles.axes_th1, 'on')

%         plot(handles.axes_th2,t(1:k), th_2(1,1:k));
% %         axis(handles.axes_th2,[0 tmax, -90 90]);
%         xlabel(handles.axes_th2,'Time(s)')
%         ylabel(handles.axes_th2,'Theta2(deg)')
%         grid (handles.axes_th2, 'on')

%         plot(handles.axes_th3,t(1:k), th_3(1,1:k));
% %         axis(handles.axes_th3,[0 tmax, -90 90]);
%         xlabel(handles.axes_th3,'Time(s)')
%         ylabel(handles.axes_th3,'Theta3(deg)')
```

```
%         grid (handles.axes_th3, 'on')
%
%        plot(handles.axes_th4,t(1:k), th_4(1,1:k));
% %         axis(handles.axes_th4,[0 tmax, -90 90]);
%        xlabel(handles.axes_th4,'Time(s)')
%        ylabel(handles.axes_th4,'Theta4(deg)')
%         grid (handles.axes_th4, 'on')

        plot(handles.axes_th1_dot,t(1:k), v_th1(1,1:k));
        xlabel(handles.axes_th1_dot,'Time(s)')
        ylabel(handles.axes_th1_dot,'Th1dot(rad/s)')
        grid (handles.axes_th1_dot, 'on')

        plot(handles.axes_th2_dot,t(1:k), v_th2(1,1:k));
        xlabel(handles.axes_th2_dot,'Time(s)')
        ylabel(handles.axes_th2_dot,'Th2dot(rad/s)')
         grid (handles.axes_th2_dot, 'on')

        plot(handles.axes_th3_dot,t(1:k), v_th3(1,1:k));
        xlabel(handles.axes_th3_dot,'Time(s)')
        ylabel(handles.axes_th3_dot,'Th3dot(rad/s)')
        grid (handles.axes_th3_dot, 'on')

        plot(handles.axes_th4_dot,t(1:k), v_th4(1,1:k));
        xlabel(handles.axes_th4_dot,'Time(s)')
        ylabel(handles.axes_th4_dot,'Th4dot(rad/s)')
         grid (handles.axes_th4_dot, 'on')
        pause(tmax/K);

    end
    x = x_1;
    y = y_1;
    z = z_1;
    pitch = pitch_1;
    [~, theta_inv_final] =
Check_Inverse_Jaco_new(x,y,z,pitch,old_theta_2,old_theta_3);
    theta1 = theta_inv_final(1,1)*180/pi;
    old_theta_1 = theta1;
    theta2 = theta_inv_final(2,1)*180/pi;
    old_theta_2 = theta2;
```

```
    theta3 = theta_inv_final(3,1)*180/pi;
    old_theta_3 = theta3;
    theta4 = theta_inv_final(4,1)*180/pi;
    old_theta_4 = theta4;
%     pause(0.001);
   DrawRobot(theta1, theta2,theta3,theta4,handles);
    hold on;
    plot3(X,Y,Z,'linewidth',2,'color','black');
    hold off;
    set(handles.slider_theta1,'Value', theta1);
    set(handles.edit_theta1,'String', num2str(round(theta1,2)));
    set(handles.slider_theta2,'Value', theta2);
    set(handles.edit_theta2,'String', num2str(round(theta2,2)));
    set(handles.slider_theta3,'Value', theta3);
    set(handles.edit_theta3,'String', num2str(round(theta3,2)));
    set(handles.slider_theta4,'Value', theta4);
    set(handles.edit_theta4,'String', num2str(round(theta4,2)));
      plot(handles.axesq,t,q);
       axis(handles.axesq,[0 tmax, 0 qmax])
       xlabel(handles.axesq,'Time(s)')
       ylabel(handles.axesq,'Positon(mm)')
       grid (handles.axesq, 'on')

      % plot(handles.axesv,t, v);
        axis(handles.axesv,[0 tmax, 0 vmax+5])
        xlabel(handles.axesv,'Time(s)')
       ylabel(handles.axesv,'Velocity(mm/s)')
       grid (handles.axesv, 'on')

       plot(handles.axesa,t, a);
        axis(handles.axesa,[0 tmax, -amax amax])
        xlabel(handles.axesa,'Time(s)')
       ylabel(handles.axesa,'Acceleration(mm/s^2)')
       grid (handles.axesa, 'on')

%        % plot(handles.axes_th1,t, th_1);
%         axis(handles.axes_th1,[0 tmax, -180 180]);
%          xlabel(handles.axes_th1,'Time(s)')
%         ylabel(handles.axes_th1,'Theta1(deg)')
%         grid (handles.axes_th1, 'on')
```

5.2. Kết quả thực hiện



6. Thành lập ma trận Jacobian và điểm kì dị

Từ các ma trận thuần nhất giữa các khớp so với base:

$$T_1^0 = \begin{bmatrix} c(\theta_1) & 0 & s(\theta_1) & 0 \\ s(\theta_1) & 0 & -c(\theta_1) & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^1 = \begin{bmatrix} c(\theta_2) & -s(\theta_2) & 0 & a_2c(\theta_2) \\ s(\theta_2) & c(\theta_2) & 0 & a_2s(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^2 = \begin{bmatrix} c(\theta_3) & -s(\theta_3) & 0 & a_3c(\theta_3) \\ s(\theta_3) & c(\theta_3) & 0 & a_3s(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^3 = \begin{bmatrix} c(\theta_4) & -s(\theta_4) & 0 & a_4c(\theta_4) \\ s(\theta_4) & c(\theta_4) & 0 & a_4s(\theta_4) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ta có: $J(q) = \begin{bmatrix} z_0 \times (p_4 - p_0) & z_1 \times (p_4 - p_1) & z_2 & z_3 \times (p_4 - p_3) \\ z_0 & z_1 & 0 & z_3 \end{bmatrix}$

J(q)

$$= \begin{bmatrix} -s_1(a_4c_{23} + a_3c_2 - a_2s_2 + a_5c_{234}) & -c_1(a_4s_{23} + a_2c_2 + a_3s_2 + a_5s_{234}) & -c_1(a_4s_{23} + a \\ c_1(a_4c_{23} + a_3c_2 - a_2s_2 + a_5c_{234}) & -s_1(a_4s_{23} + a_2c_2 + a_3s_2 + a_5s_{234}) & -s_1(a_4s_{23} + a \\ 0 & a_4c_{23} + a_3c_2 - a_2s_2 + a_5c_{234} & a_4c_{23} + a_5c \\ 0 & s_1 & s_1 \\ 0 & -c_1 & -c_1 \\ 1 & 0 & 0 \end{bmatrix}$$

Ta thấy 3 hàng cuối phụ thuộc tuyến tính nhưng cần 4 thông số để tính vận tốc góc nên giữ hàng 5 (ω_y)

J(q)

$$= \begin{bmatrix} -s_1(a_4c_{23} + a_3c_2 - a_2s_2 + a_5c_{234}) & -c_1(a_4s_{23} + a_2c_2 + a_3s_2 + a_5s_{234}) & -c_1(a_4s_{23} + a \\ c_1(a_4c_{23} + a_3c_2 - a_2s_2 + a_5c_{234}) & -s_1(a_4s_{23} + a_2c_2 + a_3s_2 + a_5s_{234}) & -s_1(a_4s_{23} + a \\ 0 & a_4c_{23} + a_3c_2 - a_2s_2 + a_5c_{234} & a_4c_{23} + a_5c \\ 0 & -c_1 & -c_1 \end{bmatrix}$$

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\\ \omega_y\end{bmatrix} = J(q)\begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \dot{\theta}_3\\ \dot{\theta}_4\end{bmatrix} \Rightarrow \begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \dot{\theta}_3\\ \dot{\theta}_4\end{bmatrix} = J(q)^{-1}\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\\ \omega_y\end{bmatrix}$$

- Nhưng ta cần phải chuyển về ma trận đại số từ ma trận hình học:

- Từ ma trận xoay Roll-Pitch-Yaw

$$R(t) = \begin{bmatrix} c\phi c\theta & c\phi s\theta & s\phi \\ s\phi c\theta & s\phi s\theta & -c\phi \\ -s\theta & c\theta & 0 \end{bmatrix} \Rightarrow R(t)^T = \begin{bmatrix} c\phi c\theta & s\phi c\theta & -s\theta \\ c\phi s\theta & s\phi s\theta & c\theta \\ s\phi & -c\phi c\theta & 0 \end{bmatrix}$$

$$\Rightarrow \dot{R}(t) = \begin{bmatrix} -s\phi c\theta\dot{\phi} - c\phi s\theta\dot{\theta} & -s\phi s\theta\dot{\phi} + c\phi c\theta\dot{\theta} & c\phi\dot{\phi} \\ c\phi c\theta\dot{\phi} - c\phi s\theta\dot{\theta} & c\phi s\theta\dot{\phi} + s\phi c\theta\dot{\theta} & s\phi\dot{\phi} \\ -c\theta\dot{\theta} & -s\theta\dot{\theta} & 0 \end{bmatrix}$$

$$S(t) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ \omega_y & \omega_x & 0 \end{bmatrix} = \dot{R}(t)R(t)^T$$

- Đồng nhất hệ số ta được

$$\omega_x = -\dot{\theta}c\theta s\phi c\theta - s\theta\dot{\theta}s\phi s\theta = -s\phi\dot{\theta}$$

$$-\omega_y = -\dot{\theta}c\theta c\phi c\theta - s\theta\dot{\theta}c\phi s\theta = -c\phi\dot{\theta} \Rightarrow \omega_y = c\phi\dot{\theta}$$

$$\omega_z = c\theta c\phi\left(c\theta c\phi\dot{\phi} - s\theta s\phi\dot{\theta}\right) + s\theta c\phi\left(s\theta c\phi\dot{\phi} + c\theta s\phi\dot{\theta}\right) + \phi s^2\phi$$

$$= \dot{\phi}(c^2\phi(c^2\theta + s^2\theta) + s^2\phi) + \dot{\theta}(s\theta c\phi c\theta s\phi - s\theta c\phi c\theta s\phi) = \dot{\phi} = \dot{\theta}_1$$

- Nên để chuyển từ ω_y sang pitch_dot ta cần phải nhân thêm cos góc yaw

- Điểm kì dị khi det $(J_{11}) = 0$

$$J_{11} = \begin{bmatrix} -s_1(a_4c_{23} + a_3c_2 - a_2s_2 + a_5c_{234}) & -c_1(a_4s_{23} + a_2c_2 + a_3s_2 + a_5s_{234}) & -c_1(a_4s_{23} + a \\ c_1(a_4c_{23} + a_3c_2 - a_2s_2 + a_5c_{234}) & -s_1(a_4s_{23} + a_2c_2 + a_3s_2 + a_5s_{234}) & -s_1(a_4s_{23} + a \\ 0 & a_4c_{23} + a_3c_2 - a_2s_2 + a_5c_{234} & a_4c_{23} + a_5c \end{bmatrix}$$

$$\det(J_{11}) = 0$$

$$\Leftrightarrow (a_4c_{23} + a_3c_2 - a_2s_2 + a_5c_{234})(a_2a_4c_3 + a_2a_5c_{34} - a_3a_4s_3 - a_3a_5s_{34}) = 0$$

- Giải phương trình:

$$a_2a_4c_3 + a_2a_5c_{34} - a_3a_4s_3 - a_3a_5s_{34} = 0$$
$$\Leftrightarrow s_3 = \frac{a_2a_4 + a_2a_5c_4 - a_3a_5s_4}{-a_2a_5s_4 - a_3a_4 - a_3a_5c_4}c_3$$
$$\Leftrightarrow \theta_3 = \arctan\left(\frac{a_2a_4 + a_2a_5c_4 - a_3a_5s_4}{-a_2a_5s_4 - a_3a_4 - a_3a_5c_4}\right)$$
$$a_4c_{23} + a_3c_2 - a_2s_2 + a_5c_{234} = 0$$
$$\Leftrightarrow s_2 = \frac{a_4c_3 + a_3 - a_5s_4c_3 + a_5c_4c_3}{-a_4s_3 - a_2 - a_5s_4c_3 - a_5c_4s_3}c_2$$
$$\Leftrightarrow \theta_2 = \arctan\left(\frac{a_4c_3 + a_3 - a_5s_4c_3 + a_5c_4c_3}{-a_4s_3 - a_2 - a_5s_4c_3 - a_5c_4s_3}\right)$$

- Nên khi giải điểm kì dị để đơn giản khi giải phương trình trên ta xét điều kiện và cả 2 tích đều bằng 0 để khi chọn được theta4 và theta1 thì ta tính được theta2 và theta3

7. Kết luận

Thông qua đề tài bài tập lớn lần này, em đã biết được những loại tay máy, robot công nghiệp khác nhau, đồng thời hiểu được những thông số cần quan tâm khi vận hành và tìm hiểu một robot công nghiệp.

Bên cạnh đó, dựa vào những hướng dẫn của thầy và những kiến thức đã được học em đã giải quyết được một số bài toán về robot như động học thuận, động học ngược, quy hoạch quỹ đạo và động học vận tốc

Từ đó có thể vận dụng và hiểu được những bài toán robot thực tế, giúp có thêm nhiều cơ hội hơn cho mình trong tương lai

Link video chạy mô phỏng và source code:

https://drive.google.com/drive/u/0/folders/1LpDZpR5Fjhe3oeKnzd8baXL0gW14zOre

8. Tài liệu tham khảo

[1] Giáo trình kỹ thuật robot, Đại học Bách khoa Đại học Quốc gia thành phố Hồ Chí Minh.

[2] Bài giảng môn Kỹ thuật robot, TS. Nguyễn Hoàng Giáp, Trường đại học Bách Khoa, đại học quốc gia thành phố Hồ Chí Minh.

[3] Robotic Modelling Planning and Control.